НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

«До захисту допущено»

Завідувач кафедри

_____ О.І. Ролік

«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 «Програмна інженерія»

на тему: « Система забезпечення взаємодії підбору та отримання товару на основі ASP.NET **»**

Виконав (-ла):
студент (-ка) IV курсу, групи ІТ-51
Черненко Максим Юрійович _____

Керівник:
Професор кафедри АУТС, д.т.н., доцент Корнієнко Б.Г. _____

Рецензент:
Доцент кафедри ОТ, к.т.н., доцент Павлов В.Г. _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.

Студент (-ка) _____

Київ – 2019 рік

Пояснювальна записка
до дипломного проекту
на тему: «Тема»

Київ – 2019 рік

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ASP.NET – платформа для розробки веб застосунків.

MVC – архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

Docker – інструментарій для управління ізольованими Linux-контейнерами.

SQL Server – система керування базами даних.

Entity framework – об'єктно-орієнтована технологія на базі фреймворка .NET для роботи з даними.

Bootstrap – це інструментарій з відкритим вихідним кодом для розробки за допомогою HTML, CSS і JS.

jQuery – бібліотека JavaScript, що фокусується на взаємодії JavaScript і HTML.

API – інтерфейс програмування, інтерфейс створення додатків.

HTML – мова розмітки гіпертекстових документів.

CSS – набір параметрів форматування, який застосовується до елементів документа, щоб змінити їх зовнішній вигляд.

JS – мова програмування.

ВСТУП

У кожній країні передбачений податок на товар, який ввозиться в країну. На сьогоднішній день це дуже не зручно для людей, які хочуть замовити якусь річь з-за кордону, тому що товар в інтернет-магазині коштує певну суму до якої додається сума доставки товару та відсотковий податок. Через це, ціна на товар може зростати майже вдвічі.

З іншого боку, мандрівник, який повертається додому з подорожі та везе з собою якусь нову техніку або подарунки, не сплачує податок. Ця ситуація підштовхнула мене на те, щоб зробити веб-додаток, за допомогою якого будь-яка людина зможе отримати свою посилку з-за кордону за допомоги мандрівників. Але головної ціллю було заохочення мандрівників, тому було вирішено заохочувати мандрівників за допомогою грошової винагороди у вигляді – десять відсотків від вартості товару.

Отже, завданням дипломного проекту було створення системи управління, яка буде управляти та контролювати взаємодію між людиною – замовником, яка підбирає товар, та людиною – виконавцем, яка доставляє товар, підібраний замовником. Тому переді мною постала непроста задача розробити чіткий алгоритм взаємодії користувачів. Спочатку користувач повинен зареєструватись у веб-додатку вказавши такі дані як – ім’я, прізвище, електронна адреса та інші. Після реєстрації користувачі можуть створювати новий заказ або вибрати один із заказів для виконання. Користувач, якому потрібен продукт з-за кордону, подає заявку. Йому потрібно вказати ціну покупки, бажаний час доставки, країну звідки товар та країну в якій вій забере товар. В ідеалі кожне «замовлення» повинно супроводжуватися детальним описом того, де і як можна придбати потрібний товар (посилання на інтернет-магазин або точну адресу). В іншому випадку (якщо мандрівнику необхідно самостійно шукати, де і як знайти товар), йому доведеться більше платити за послуги доставки через витрачений час. Мандрівники, які можуть доставити ту чи іншу річ, починають залишати відгук із зазначенням дати прибуття. Після прибуття «виконавця» товари переводяться

«замовнику» в узгодженому порядку. Це може бути як особиста зустріч, так і поштова посилка.

Тепер головне - про розмір і спосіб оплати. Немає конкретної ціни на послуги, однак, за статистикою, плата за таку «доставку» становить близько 10% від ціни товару. Процес оплати на етапі розробки буде проблемним - треба буде вирішити, як найбільш зручно його реалізувати. Якщо клієнт здійснює передоплату - мандрівник може просто підняти його і "розділити". І навпаки - якщо мандрівник купує товари за свої гроші без гарантій - посилка може бути не отримана після прибуття.

Для захисту обох сторін послуга буде здійснена таким чином, що при замовленні ціна покупки знімається з картки покупця. Зарядка та переказ на картку здійснюється після того, як клієнт підтвердить отримання посилки. Переказ на картку здійснюється після того, як клієнт підтвердить отримання посилки.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Програмний продукт, що задовольняє вимогам повинен бути написаний на мові, що дозволяє запуск на операційній системі macOS. Він повинен підтримувати HTTP запити та кодування даних за допомогою formdata. Сайт повинен бути розроблений за технологією ASP.NET. Як сервер бази даних використовують Microsoft SQL Server. Клієнтська частина сайту повинна бути доступною широкого кола користувачів. Ви повинні створити клієнтську та адміністративну частини. Також повинні бути розроблені зрозумілі та зручні інтерфейси.

1.2 Змістовний опис і аналіз предметної області

Система управління повинна бути побудована на основі клієнт-серверної архітектури. Ця технологія дозволяє зв'язати декілька систем між собою. Для реалізації слід використовувати мову програмування C# та шаблон MVC. Реалізувати це за допомогою платформи для розробки ASP.NET. Шаблон MVC, що лежить в основі ASP.NET платформи, передбачає взаємодію трьох компонентів: контролера (controller), моделі (model) і перегляду (view). Для зберігання та обміну даними треба підключити базу даних SQL Server. Щоб її запустити та працювати з нею в реальному часі потрібно використати потужний інструмент Docker.

1.3 Аналіз успішних IT-проектів

На сьогодняшній день у світі є декілька існуючих сервісів зі схожим функціоналом:

1.3.1 Travelpost - https://travelpost.com

Цей сервіс дозволяє знайти мандрівника, який відправляється в потрібне місто і "орендувати місце в валізі", пропонуючи нагороду за перевезення вантажів. На даний момент цей сервіс повністю безкоштовний, вартість послуг пропонує сам замовник. Завдяки такому рішенню, з'явилася дуже велика конкуренція від виконавців, тому що вони змагаються між собою, пропонуючи нижче ціну сплати за доставку, щоб виконати цей заказ.

У цьому додатку присутній рейтинг рейтинг користувача, його виставляють інші користувачи, після того, як відбувся обмін посилками. Оцінка залежить від швидкості виконавця, ціни за доставку та якості доставки.

1.3.2 Бандеролька - https://qwintry.com

Бандеролька – це веб-додаток за допомогою якого можна доставити покупки з Америки зручно та надійно. Це зручно тоді, коли в інтернет-магазині немає можливості доставити товар в вашу країну. Коли ви хочете замовити декілька речей з Америки, ви в інтернет-магазині вказуєте адресу доставки ту, яку вам вкажуть на сервісі «Бандеролька», потім всі ваші замовлення формуються в одне та складаються в одну посилку. Згодом вони відправляються в вашу країну за допомогою різних почтових сервісів.

Сервіс гарантує повернення всієї вартості вашої посилки, якщо з нею щось станеться. Також у «Бандеролькі» є багато обмежень щодо виду товару, який доставляється. Сервіс не може перевезти вибухонебезпечні речовини, легкозаймисті речовини, зброю та багато іншого. Якщо ви замовили товар на адресу, яку ви отримали від бандеролькі, оплатили його, а згодом виявилося що, це – легкозаймиста речовина, вами буде втрачено ту сумму, яку ви витратили на цей товар, тому що зазвичай інтернет-магазини Америки не повертають легкозаймисті товари, вибухонебезпечні, та інші. Але про це вам

ніхто не повідомить заздалегідь, лише після того, як ваш товар буде доставлено на сервіс «Бандеролька».

1.3.3 Grabr - https://grabr.io

Робота програми базується на двох типах людей: клієнтах і мандрівниках. У цьому випадку в різний час кожен користувач може бути обома. Все працює дуже просто: покупець вкладає в додаток продукт, який хоче отримати, і вибирає певну суму винагороди - це може бути або $ 5, або $ 500. Ціна, але багато що залежить від розміру товару і наскільки важко його принести: хтось попросить айфон, який можна покласти в ручний багаж, а хтось - кермо для велосипеда. Подорожувачі заздалегідь вводять дату і напрямок поїздки, тому вони отримують повідомлення про ті замовлення, які чекають від місця свого перебування. Розглянувши пропозицію, він може або прийняти його, або погодитися на іншу суму. У той же час, після того, як мандрівник прийняв замовлення, узгоджена сума блокується на картці клієнта. Таким чином, мандрівник може бути впевнений, що отримає ці гроші, якщо він його доставить, а покупець, що його гроші не будуть списані, поки він не отримає товар.

Ідея Grabr дуже сподобалася багатьом, але користувачеві не так просто дійти до замовлення. Часто він хоче спробувати щось купити, але він не знає, що. Тому Grabr підштовхує клієнтів до покупки - на головній сторінці програми є вибір найпопулярніших продуктів: всякі гаджети, різні кросівки і, звичайно, їжа. Сервіс також працює над залученням громадськості до платформи, яка багато подорожує і знає цікаві дивовижні речі по всьому світу. Це реалізовано з допомогою блогу в якому часто постять інформацію, таку як « Що замовити з Кіпру?» та подібне.

Сервіс за свою роботу с кожного замовлення отримує 7% від вартості замовлення.

1.3.4 Gransjoy - https://www.gransjoy.com

Gransjoy - це сервіс, який дозволяє відправляти посилки в будь-яку точку світу або замовляти предмети з-за кордону за допомогою мандрівників. Ті, хто замовляють – економлять, а мандрівники роблять добру справу і отримують приємні бонуси. Наприклад, ви можете попросити мандрівника принести тайський ром або синій чай, а натомість підвести його з аеропорту. На Gransjoy, мандрівники реєструють свої маршрути, вказуючи дати та пункти відправлення та призначення. Ви можете зайти на сайт, знайти відповідного мандрівника, написати йому в соціальних мережах або зв'язатися з ним по електронній пошті і попросити привести вам екзотичний фрукт, або, можливо, ви повинні відправити подарунок другу або документи до вашої сестри.

ВИСНОВКИ ПО РОЗДІЛУ

Під час проведення аналізу вимог до програмного забезпечення було розглянуто та проаналізовано предметну область – веб додатки на основі клієнт-серверної архітектури. Також, були оглянуті успішні існуючі технічні рішення та порівняння поставлених задач з відомими веб додатками.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Розроблена система виконується операційними системами на базі MacOS. Можливе використання під операційними системами Linux та Windows, але не тестувалось.

Структура сайту - систематизація інформації та навігація по ній з метою адаптувати відвідувачів до більш успішного пошуку потрібної їм продукції. Добре продумана структура сайту гарантує, що користувачі витратять менше часу на пошук потрібного інструменту. Розробка структури сайту здійснюється з урахуванням найбільш важливої інформації. з точки зору просування товарів або послуг на Інтернет-ринку. В процесі створення структури сайту або оптимізації структури існуючого, необхідно зосередити увагу користувачів саме на тій самій інформації розділи сайту, які є найбільш важливими відповідно до позиціонування ринок, товари або послуги.

Для розробки браузерної частини системи було застосовано мова розмітки HTML5, з використанням фреймворку Bootstrap та таблиці стилів CSS. Для належної роботи фраємворка bootstrap було підключено бібліотеку jQuery. Bootstrap робить фронтенд веб-розробку швидше і простіше. Це зроблено для людей з будь-яким рівнем підготовки, пристроїв всіх форм і проектів будь-якого масштабу. Bootstrap використовує новітні можливості HTML і CSS. Вони можуть не підтримуватися старими версіями браузерів. В першу чергу будьте обережні з IE7 і 8. Починаючи з версії Bootstrap 2 всі макети створювані з його допомогою стали адаптивними[1]. jQuery - це швидка, невелика та багатофункціональна бібліотека JavaScript. Вона робить такі речі, як обхід документів HTML та маніпуляції, обробка подій, анімацію та Ajax набагато простіше з використанням простих у використанні API, які працюють у багатьох браузерах. Завдяки поєднанню універсальності та розширюваності jQuery змінив

спосіб, у який мільйони людей пишуть JavaScript. Будь-хто, хто написав серйозний JavaScript в минулому, може засвідчити, що невідповідності між браузерами змусять вас зненацька. Наприклад, дизайн, який відмінно відображається в Mozilla Firefox і Internet Explorer 8, просто розвалюється в Internet Explorer 7, або інтерфейсний компонент, який ви проводили багато днів, прекрасно працює у всіх основних браузерах, крім Opera на Linux. А клієнт просто використовує Opera на Linux. Ці типи проблем ніколи не можна легко відстежити, і навіть важче повністю викорінити їх[2].

Використано паттерн MVC який дозволяє зручно та швидко додавати новий функціонал та відокремлювати різні частини коду, оскільки MVC дозволяє відокремити дані від бізнес-логіки та від логіки відображення. За рахунок такого поділу підвищується можливість повторного використання програмного коду і спрощується супровід (зміни зовнішнього вигляду, наприклад, не відображаються на бізнес-логіці).

Концепція MVC розділяє дані, подання та обробку дій користувача на компоненти:

– модель (Model) - надає собою об'єктну модель якоїсь предметної області, включає в себе дані і методи роботи з цими даними, реагує на запити з контролера, повертаючи дані або змінюючи свій стан, при цьому модель не містить в собі інформації, як дані можна візуалізувати, а також не "спілкується" з користувачем безпосередньо;

– представлення (View) - відповідає за відображення інформації (візуалізацію), одні і ті ж дані можуть представлятися різними способами, наприклад, колекцію об'єктів за допомогою різних «вьюха» можна уявити як в табличному вигляді, так і списком;

– контролер (Controller) - забезпечує зв'язок між користувачем і системою, використовує модель і уявлення для реалізації необхідної реакції на дії користувача, як правило, на рівні контролера здійснюється фільтрація

отриманих даних і авторизація (перевіряються права користувача на виконання дій або отримання інформації)[3].

Для розробки серверної частини було застосовано мову C#. Для роботи з базою даних використано Entity Framework. Entity Framework являє спеціальну об'єктно-орієнтовану технологію на базі фреймворка .NET для роботи з даними. Якщо традиційні засоби ADO.NET дозволяють створювати підключення, команди та інші об'єкти для взаємодії з базами даних, то Entity Framework являє собою більш високий рівень абстракції, який дозволяє абстрагуватися від самої бази даних і працювати з даними незалежно від типу сховища. Якщо на фізичному рівні ми оперуємо таблицями, індексами, первинними і зовнішніми ключами, але на концептуальному рівні, який нам пропонує Entity Framework, ми вже працюємо з об'єктами. Центральної концепцією Entity Framework є поняття сутності або entity. Сутність представляє набір даних, асоційованих з певним об'єктом. Тому дана технологія передбачає роботу не з таблицями, а з об'єктами і їх наборами. Будь-яка сутність, як і будь-який об'єкт з реального світу, має низку властивостей. Наприклад, якщо сутність описує людини, то ми можемо виділити такі властивості, як ім'я, прізвище, зріст, вік, вага. Властивості необов'язково представляють прості дані типу int, а й можуть представляти більш комплексні структури даних. І у кожної сутності може бути одна або кілька властивостей, які будуть відрізняти цю сутність від інших і будуть унікально визначати цю сутність. Подібні властивості називають ключами. При цьому суті можуть бути пов'язані асоціативної зв'язком один-ко-многим, один-ко-одному і багато-до-багатьох, подібно до того, як в реальній базі даних відбувається зв'язок через зовнішні ключі. Відмінною рисою Entity Framework є використання запитів LINQ для вибірки даних з БД. За допомогою LINQ ми можемо не тільки отримувати певні рядки, що зберігають об'єкти, з бд, а й отримувати об'єкти, пов'язані різними асоціативними зв'язками. Іншим ключовим поняттям є Entity Data Model. Ця модель зіставляє класи сутностей з реальними таблицями в БД [4].

Для зберігання даних було використано базу даних Microsoft SQL Server, для її коректної роботи було застосовано платформу Docker. Microsoft SQL Server - одна з найбільш потужних систем роботи з базами даних в архітектурі "клієнт-сервер". Особливість системи - робота сервера тільки в операційних системах ряду Microsoft Windows NT - NT Server 4.0, 2000 Server, Server 2003, при цьому клієнтська частина може взаємо-діяти з сервером з Microsoft Windows 98 і інших операційних систем. Рекомендована файлова система для SQL Server - NTFS, хоча можлива робота і в системі FAT. У своєму складі система має засоби створення баз даних, роботи з інформацією баз даних, перенесення даних з інших систем і в інші системи, резервного копіювання та відновлення даних, розвинену систему транзакцій, систему реплікації даних, реляційну підсистему для аналізу, оптимізації та виконання запитів клієнтів , систему безпеки для управління правами доступу до об'єктів бази даних. Система не містить засобів розробки клієнтських додатків[5].

Docker - це відкрита платформа для розробки, доставки і експлуатації додатків. Docker розроблений для більш швидкого викладання ваших додатків. За допомогою docker ви можете відокремити ваше додаток від вашої інфраструктури і звертатися з інфраструктурою як керованим додатком. Docker допомагає викладати ваш код швидше, швидше тестувати, швидше викладати додатки і зменшити час між написанням коду і запуску коду. Docker робить це за допомогою легковагій платформи контейнерної віртуалізації, використовуючи процеси і утиліти, які допомагають керувати і викладати ваші програми.

У своєму ядрі docker дозволяє запускати практично будь-який застосунок, безпечно ізольоване в контейнері. Безпечна ізоляція дозволяє вам запускати на одному хості багато контейнерів одночасно. Легка природа контейнера, який запускається без додаткового навантаження гипервизора, дозволяє вам добиватися більше від вашого заліза. Docker прекрасно підходить для організації циклу розробки. Docker дозволяє розробникам використовувати локальні

контейнери з додатками і сервісами. Що надалі дозволяє інтегруватися з процесом постійної інтеграції та викладання (continuous integration and deployment workflow)[6].

Опис класів наведено у таблиці 2.1. Більш детальний опис методів наведено у таблиці 2.2. Структурні схеми класів та порядку виконання наведено у додатку Х.

Таблиця 2.1 – Опис класів (структур) системи

Клас (структура)	**Опис**
Authorization	Відповідає за авторизацію в залежності від ролі.
Dashboard	Відповідає за особистий кабінет користувача або адміна
HomeController	Відповідає за головну сторінку
Registration	Створює нового користувача у системі
ApplicationContext	Поєднує базу даних та моделі
Login	Відповідає за логін користувача
Message	Відповідає за обмін повідомленнями між користувачами
NewOrderUrl	Починає створення нового замовлення за посиланням

Order	Збирає інформацію про замовлення та створює замовлення
User	Модель користувача
RouteConfig	Налаштовое вірну компановку файла
WebApiConfig	Налаштовує API

Продовження таблиці 2.1

Клас (структура)	Опис
Authorization/Index	Сторінка авторизації
Dashboard/Admin	Особиста сторінка адміністратора
Dashboard/Client	Особиста сторінка замовника
Dashboard/Courier	Особиста сторінка курьера
Dashboard/NewMessage	Сторінка для відправки нового повідомлення
Home/CompleteOrder	Сторінка с заказами для курьера
Home/Index	Головна сторінка сайту
Home/NewOrder	Сторінка для додавання посилання на товар
Home/NewOrderFinal	Сторінка з формою відправки даних про товар
Registration/Index	Сторінка реєстрації

Таблиця 2.2 – Опис методів класів та інтерфейсів системи

Клас/Інтерфейс	Метод	Опис
Authorization	Index()	Повертає представлення
Authorization	Logout()	Деавторизує користувача
Authorization	Index(Login inputModel)	Авторизує користувача
Dashboard	Index()	Завантажує особисту сторінку користувача
Dashboard	Client()	Завантажує особисту сторінку клієнта
Dashboard	Courier()	Завантажує особисту сторінку курьера
Dashboard	CourierCompleteOrder(int id)	Додає замовлення курьеру при підтвердженні
Dashboard	ClientConfirmOrder(int id)	Відповідає за підтверження замовлення кліентом
User	Role	Перевіряє роль користувача

Продовження таблиці 2.2

Клас/Інтерфейс	Метод	Опис
Dashboard	AdminAllowOrder(int id)	Перевірка замовлення адміністратором
Dashboard	NewMessage(int id)	Повертає нове повідомлення по id клієнта
Dashboard	NewMessage(Message msg)	Створює нове повідомлення для замовника або курьера
Dashboard	Admin()	Особиста сторінка для адміністратора
HomeController	Index()	Повертає представлення
HomeController	NewOrder()	Повертає представлення
HomeController	NewOrder(NewOrderUrl inputModel)	Валідація та перехід за посиланням
User	SetPassword(string pass)	Зберігає пароль користувача та зв'язок БД, валідація

Продовження таблиці 2.2

Клас/Інтерфейс	Метод	Опис
HomeController	NewOrderFinal(Order inputModel)	Валідація та перехід за посиланням
HomeController	CompleteOrder()	Повертає представлення
HomeController	NewOrderFinal()	Повертає представлення
Registration	Index()	Повертає представлення
Registration	Index(User inputModel)	Реєструє користувача
ApplicationContext	base("ApplicationContext")	Відповідає за створення таблиць в БД
ApplicationContext	DbSet<User>	Відповідає за дані користувача в БД
ApplicationContext	DbSet<Order>	Відповідає за замовлення в БД

Продовження таблиці 2.2

Клас/Інтерфейс	Метод	Опис
ApplicationContext	DbSet<Message>	Знаходить репозиторій у базі даних по ідентифікатору.
Login	Email { get; set; }	Валідація електронної адреси та перевірка с БД
Login	Password { get; set; }	Валідація паролю та перевірка с БД
Message	Id { get; set; }	З'єднує повідомлення та унікальний id
Message	FromUserId { get; set; }	З'єднує повідомлення та id відправника
Message	ToUserId { get; set; }	З'єднує повідомлення та id отримувача
Message	Text { get; set; }	Відповідає за текст повідомлення
User	CheckPassword(string pass)	Перевіряє пароль користувача та зв'язок БД, валідація

Продовження таблиці 2.2

Клас/Інтерфейс	Метод	Опис
Message	SentDate { get; set; }	Відповідає за відправку повідомлення
NewOrderUrl	Url { get; set; }	Валідація та відправка даних, перехід за посиланням, валідація
Order	Id { get; set; }	Відповідає за унікальний id замовлення та відправку до БД, валідація
Order	Name { get; set; }	Відповідає за назву замовлення та відправку до БД, валідація
Order	Url { get; set; }	Відповідає за посилання замовлення та відправку до БД, валідація

Order	Price { get; set; }	Відповідає за ціну та відправку до БД, валідація

Продовження таблиці 2.2

Клас/Інтерфейс	**Метод**	**Опис**
Order	DeliveryDeadline { get; set; }	Відповідає за дату отримання та відправку до БД, валідація
Order	FromCountry { get; set; }	Відповідає за країну товару та відправку до БД, валідація
Order	OrderedByUserId { get; set; }	З'єднує замовника та замовлення
Order	DeliveredByUserId { get; set; }	З'єднує кур'єра та замовлення
Order	Status { get; set; }	Відповідає за статус замовлення(на перевірці, в обробці, доставлено)
Order	Address { get; set; }	Відповідає за країну товару та

		відправку до БД, валідація
Order	OrderStatus	Відповідає за поточний стан замовлення

Продовження таблиці 2.2

Клас/Інтерфейс	**Метод**	**Опис**
User	Id { get; set; }	Відповідає за унікальний id користувача та зв'язок БД, валідація
User	Email { get; set; }	Відповідає за електронну адресу користувача та зв'язок БД, валідація
User	FirstName { get; set; }	Відповідає за ім’я користувача та зв'язок БД, валідація
User	LastName { get; set; }	Відповідає за прізвище користувача та зв'язок БД, валідація
User	Balance { get; set; }	Відповідає за баланс користувача
User	Role { get; set; }	Відповідає за роль користувача

User	Password { get; set; }	Відповідає за пароль користувача та зв'язок БД, валідація
User	PasswordHash { get; set; }	Завантажує пароль користувача
RouteConfig	RegisterRoutes(RouteCollection routes)	Маршрутизація вхідних URL

Продовження таблиці 2.2

Клас/Інтерфейс	**Метод**	**Опис**
Authorization/Index	Html.ActionLink	Посилання на сторінці авторизації
Authorization/Index	Html.ValidationSummary()	Відображення помилок валідації на сторінці авторизації
Authorization/Index	Html.TextBox	Поле для заповнення інформації
Authorization/Index	Html.ValidationMessage("Email")	Відображення помилок валідації на сторінці авторизації
Dashboard/Admin	Html.ActionLink	Посилання на сторінці адміністратора .
Dashboard/Admin	order.Name	Відповідає за список с замовленнями на

		сторінці адміністратора
Registration/Index	Html.RadioButton	Відповідає за вибір ролі на сторінці реєстрації
Registration/Index	Html.TextBox	Поле для заповнення інформації

У наведених вище таблицях описано класи, їх методи та короткий опис.

2.2 Аналіз безпеки даних

Паролі користувачів у базі зберігаються у вигляді результату обробки функцією argon2, саме тому буде досить важко підібрати пароль завдяки хешу bcrypt. Argon2 використовує незалежний від даних доступ до пам'яті. Він повільніше, тому що він робить більше проходів по пам'яті для захисту від компромісів. Це настійливо рекомендується для хешування паролів і виведення пароля на основі пароля. Інформація, яка отримується у веб файлі cookie перевіряється за допомогою секретного ключа.

ВИСНОВОК ПО РОЗДІЛУ

У цьому розділі було розроблено структуру веб додатку, проаналізовано систему захисту бази даніх. Також були виписані усі класи та методи в таблиці та повністю описані їхні дії.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Аналіз статичного коду є методом аналізу якості програмних продуктів, протягом яких аналізований продукт не запускається і не виконується. Для такого аналізу використовується спеціалізоване програмне забезпечення, що дозволяє обробляти програмне забезпечення, написане на десятках різних мов програмування, виявляючи в ньому помилки, рекомендуючи правильний дизайн коду, а також підраховуючи кількість показників або метрик, які визначають якість програмного забезпечення. Насправді, цей метод є автоматизованою версією огляду коду, одним з найбільш ефективних, але в той же час найдорожчих методів виявлення помилок у програмі. Використовуючи статичний аналіз коду, можна значно здешевити процес виявлення і подальшого усунення помилок в програмному коді, в тому числі і великого списку проблем безпеки. [7].

Моделі якості ПО мають наступні чотири рівні уявлення. Перший рівень відповідає визначенню характеристик (показників) якості ПО, кожна з яких відображає окрему точку зору користувача на якість. Згідно зі стандартами ДСТУ ISO / IEC 9126-93, ДСТУ ISO / IEC 12119-2000, ГОСТ 28195-89, в модель якості входить шість характеристик, або шість основних показників якості, які перерахуємо в порядку їх значимості для більшості користувачів:

- функціональні можливості;
- функціональна надійність;
- зручність застосування;
- ефективність;
- сопровождаемость;
- переносимість.

Функціональні можливості - сукупність властивостей, що визначають здатність програмного забезпечення виконувати передбачені функції в заданому середовищі відповідно до заданих вимог. Під функцією розуміється деяка упорядкована послідовність дій для задоволення споживчих властивостей. Функції бувають цільові (основні) і допоміжні.

До атрибутів функціональних можливостей відносяться:

– захищеність - атрибут, який показує здатність ПО запобігати несанкціонований доступ (випадковий або умисний) до програм і даних;

– інтероперабельність - атрибут, який показує можливість взаємодії даного ПЗ зі спеціальними системами і середовищами (операційні системи, обчислювальні мережі);

– функціональна повнота - атрибут, який показує міру достатності основних функцій для вирішення завдань відповідно до призначення даного ПЗ.

Функціональна надійність. До атрибутів функціональної надійності ПО відносяться:

- безпомилковість - атрибут, який визначає здатність ПО функціонувати без помилок;
- правильність - атрибут, який показує міру досягнення правильних результатів;
- контрольованість - атрибут, який характеризує повноту і ефективність виявлення помилок в проміжних і вихідних результатах;
- стійкість до помилок - атрибут, який характеризує здатність ПЗ правильно виконувати функції при аномальних умовах (збій апаратури, помилки в даних і інтерфейсів і ін.);
- безвідмовність - атрибут, який характеризує здатність ПЗ не викликати функціональні відмови інформаційної системи;
- придатність до відновлення - атрибут, який характеризує здатність програми до усунення програмної помилки і до перезапуску для

повторного виконання і відновлення даних в разі функціонального відмови;

– готовність - атрибут, який показує здатність програми по довільній заявці безпомилково виконати передбачене перетворення.

Зручність застосування характеризується великою кількістю атрибутів, які вказують на необхідні і придатні умови використання ПЗ заданим колом користувачів для отримання відповідних результатів.

До атрибутів зручності застосування відносяться:

• розуміння - атрибут, який визначає зусилля, витрачені на розпізнавання логічних концепцій та умов застосування ПО;

• вивчаємість (легкість вивчення) - атрибут, який визначає зусилля користувачів, спрямовані на визначення застосовності ПО шляхом використання операційного контролю, діагностики, а також процедур, правил і документації;

• оперативність - атрибут, який показує реакцію системи при виконанні операцій і операційного контролю.

Ефективність - сукупність атрибутів, які визначають взаємозв'язок рівнів виконання ПО, використання ресурсів (кошти, апаратура, матеріали - папір для друкувального пристрою і ін.) І послуг, які виконуються штатним обслуговуючим персоналом та ін.

До атрибутів ефективності ПО відносяться:

– реактивність - атрибут, який показує час відгуку, обробки і виконання функцій;

– ефективність ресурсів - атрибут, що показує кількість і тривалість використовуваних ресурсів при виконанні функцій ПО;

– узгодженість - атрибут, який показує відповідність даної характеристики заданим стандартам, правилам і приписам.

Супровід - сукупність властивостей, які характеризують зусилля, які треба затратити на проведення модифікацій, що включають коригування,

вдосконалення та адаптацію ПО при зміні середовища, вимог або функціональних специфікацій.

Супровід включає наступні атрибути:

– аналізіруемость- атрибут, який визначає необхідні зусилля для діагностики відмов або ідентифікації частин, які будуть модифікуватися;

– стабільність - атрибут, який вказує на сталість структури і ризик її модифікації;

– тестованих -атрібут, який вказує на зусилля при проведенні валідації та верифікації з метою виявлення невідповідностей вимогам, а також на необхідність проведення модифікації ПО і сертифікації;

– змінність - атрибут, який визначає можливість видалення помилок в ПЗ або внесення змін для їх усунення, а також введення нових можливостей в ПО або в середу функціонування.

Переносимість - безліч показників, що вказують на здатність ПО адаптуватися до роботи в нових умовах середовища виконання. Середовище може бути організаційної, апаратної і програмної. Тому перенесення ПО в нове середовище виконання може бути пов'язаний з сукупністю дій, спрямованих на забезпечення його функціонування в середовищі, відмінному від того середовища, в якій воно створювалося, з урахуванням нових програмних, організаційних і технічних можливостей.

Переносимість включає атрибути:

– адаптивність - атрибут, що визначає зусилля, витрачені на адаптацію до різних середовищ;

– зручні налаштування (простота інсталяції) - атрибут, який визначає необхідні зусилля для запуску даного ПЗ в спеціальному середовищі;

– співіснування - атрибут, який визначає можливість використання спеціального ПЗ в середовищі діючої системи;

– замінність - атрибут, який характеризує можливість перенесення ПО з одного інструментального середовища в іншу з необхідною інсталяцією або адаптацією ПО.

Для оцінки атрибутів якості на етапах життєвого циклу (при перегляді документації, програм і результатів тестування програм) використовуються метрики з заданим оцінним вагою для нівелювання результатів метричного аналізу сукупних атрибутів конкретного показника і якості в цілому. Атрибут якості визначається за допомогою однієї або декількох методик оцінки на етапах

Залежно від призначення, особливостей і умов супроводу ПО, вибираються найбільш важливі характеристики якості і їх атрибути. Вибрані атрибути і їх пріоритети відображаються у вимогах на розробку систем, або використовуються відповідні пріоритети еталона класу ПО, до якого це ПЗ належить. [8].

3.2 Підходи до тестування

Для тестування будуть використані наступні методи:

– компонентне;

– системне;

– продуктивності.

3.2.1 Компонентне тестування

Компонентне (модульне) тестування перевіряє функціональність і шукає дефекти в частинах додатка, які доступні і можуть бути протестовані по-окремо (модулі програм, об'єкти, класи, функції і т.д.). Зазвичай компонентне (модульне) тестування проводиться викликаючи код, який необхідно перевірити і при підтримці середовищ розробки, таких як фреймворки для модульного тестування або інструменти для налагодження.

Основна ідея модульного тестування полягає в тому, щоб написати тести, в яких перевірена найменша «одиниця» коду. Модульні тести зазвичай написані на тій же мові програмування, що і вихідний код програми. Вони створюються безпосередньо для перевірки цього коду. Тобто модульні тести - це код, який перевіряє коректність іншого коду. Слово «тест» в контексті я використовую досить ліберально, тому що модульні тести в якомусь сенсі тестами не є. Вони нічого не відчувають. Я маю на увазі, що при запуску модульного тесту ви зазвичай не виявляєте, що якийсь код не працює. Ви це виявляєте під час написання тесту, оскільки ви будете міняти код до тих пір, поки тест не стане зеленим. Так, код може змінитися пізніше, і тоді ваш тест може потерпіти невдачу. Так що в цьому сенсі модульний тест є регресійний тестом. Модульний тест не схожий на звичайний тест, де у вас є кілька кроків, які ви збираєтеся виконати, і ви бачите, чи працює програмне забезпечення правильно чи ні. В процесі написання модульного тесту ви виявляєте, чи робить код то, що він повинен чи ні, і будете міняти код до тих пір, поки тест не буде пройдено. Можна вважати модульний тест абсолютної специфікацією. Модульний тест визначає, що в цих умовах, з цим конкретним набором вхідних даних, є результат, який ви повинні отримати від цього модуля коду. Істинне модульне тестування дозволяє визначити найменшу зв'язну одиницю коду, яка в більшості мов програмування - по крайней мере, об'єктно-орієнтованих - є класом[9].

Методом компонентного тестування будуть перевірені логічно окремі частини коду та методи, такі як:

– методи доступу до бази даних;

– окремі допоміжні функції;

3.2.2 Системне тестування

Системне тестування призначене для тестування готового ПО в тому стані, в якому воно буде впроваджуватися в дослідно-промислову експлуатацію.

Системне тестування дозволяє виявити такі дефекти як виявлення відсутнього функціоналу в системі, некоректна робота функцій системи, виникнення помилок при використанні специфічних тестових даних або їх комбінації, у разі помилки зв'язку з іншими системами. истемне тестування якісно відрізняється від інтеграційного і модульного рівнів. Системне тестування розглядає тестируемую систему в цілому і оперує на рівні користувача інтерфейсів, на відміну від останніх фаз інтеграційного тестування, яке оперує на рівні інтерфейсів модулів. Різні й цілі цих рівнів тестування. На рівні системи часто складно і малоефективно аналізувати проходження тестових траєкторій усередині програми або відслідковувати правильність роботи конкретних функцій. Основне завдання системного тестування - у виявленні дефектів, пов'язаних з роботою системи в цілому, таких як неправильне використання ресурсів системи, непередбачені комбінації даних користувача рівня, несумісність з оточенням, непередбачені сценарії використання, відсутня або неправильна функціональність, незручність в застосуванні тощо. У цілому системне тестування проводиться на користувальницьких інтерфейсах, створюється ілюзія того, що створено спеціальну систему автоматизації тестування не завжди необхідно. Однак обсяг даних на такому рівні, що зазвичай є більш ефективним, є повною або частичною автоматизацією тестування, що призводить до створення тестової системи.

Системне тестування проводиться над проектом в цілому за допомогою методу «чорного ящика». Структура програми не має ніякого значення, для перевірки доступні тільки входи і виходи, видимі користувачеві. Тестуванню підлягають коди і призначена для користувача документація[10].

Було протестовано наступні модулі:

- робота сервера та замовника з базою даних;
- робота сервера та виконавця з чергою подій;
- взаємодія замовника та виконавця;

– взаємодія адміна та сервісу;

3.2.3 Тестування продуктивності

Вважається, що тестування продуктивності - це те тестування, яке не є функціональним. Існує безліч видів тестування продуктивності. Класифікація видів тестування продуктивності будується на основі того, які цілі переслідує певний вид тестування. Як правило тестування продуктивності переслідує не одну, а кілька цілей в зв'язку з тим, багато типів тестування в ході його проведення поєднуються з іншими цілями або повторюються кілька разів в ході циклу тестування. Основна відмінність тестування продуктивності також полягає в тому, що воно відбувається тільки після повного функціонального тестування. Помилки функціональністю не виправляються в ході тестування продуктивності. Для даного виду тестування найчастіше виділяється окремий навантажувальний стенд, що повторює копію промислового стенду. Тестування навантаження (load testing) - даний тип тестування дозволяє оцінити поведінку системи при зростаючій навантаженні, метою навантажувального тестування є також визначення максимального навантаження, яке може витримати системах[11].

Методом тестування продуктивності буде перевірена швидкодія наступних елементів системи:

– запити до бази даних;

– обмін повідомленнями між користувачами

– паралельне виконання декількох процесів.

3.3 Критерії проходження тестування

Почнемо з необхідних умов. Очевидно, що тестування не можливо без об'єкта тестування, звідси отримуємо перша умова: наявність об'єкта тестування,

доступного для проведення випробувань. Далі, щоб тестування все ж відбулося, нам потрібен виконавець, значить, другим необхідною умовою буде наявність виконавця, причому їм може бути, людина або машина, або комбінація людина + машина. Перейдемо до визначення достатніх умов. Бо ж не кожна дія, вчинене над програмою, з метою отримання очікуване поведінки, є тестуванням, постало питання про те, що сама по собі за мету, а саме: тестування, є одним з достатніх умов. Беручи за основу те, що наявність плану тестування прямо вказує на намір проведення тестування, отримуємо, що тест план є одним з достатніх умов.

3.3.1 Компонентне тестування

Для компонентного тестування критерієм проходження є успішне виконання кожного пункту тесту. У разі якщо хоча б один пункт не був успішно виконаний – тестування вважається не пройденим.

3.3.2 Системне тестування

Для інтеграційного тестування критерієм проходження є успішне виконання кожного пункту тесту. У разі якщо хоча б один пункт не був успішно виконаний – тестування вважається не пройденим.

3.3.3 Тестування продуктивності

Для тестування продуктивності критерієм проходження є успішне виконання тесту з кожним доступним набором параметрів (кількість даних у одному запиті, кількість запитів, конкурентність) не довше ніж максимально допустимий час. У разі якщо хоча б один варіант тесту не був успішним або виконувався довше максимально допустимого часу – тестування вважається не пройденим.

3.4 Критерії проходження тестування

3.4.1 Дані до тестів

Вхідними даними для компонентного тестування є набори параметрів на яких очікується певний результат, що є вихідними даними даного тесту.

Вхідними даними для системного тестування є набори повідомлень що будуть передані від одного компоненту системи до іншого відповідно до конкретного тесту. Вихідними даними для даного виду тестування є результат роботи останнього компоненту у ланцюзі.

Вхідними даними до тестування продуктивності є набори даних, що покривають усі варіанти роботи системи у конкретному випадку. Вихідними даними є швидкість обробки запитів, кількість оброблених запитів, кількість неправильних реакцій на набір даних, дані по навантаженню на апаратну платформу.

3.4.2 Задачі тесту

Кожен тест повинен перевірити як правильність програми у відповідності до умов виконання тесту (test-driven development), так і виявити можливі помилки у роботі.

3.4.3 План виконання

Компонентне тестування викоюється у першу чергу, після цього тестування починає роботу тестування системи, та у кінці тестування продуктивності

3.5 Вимоги до середовища

3.5.1 Апаратна частина

Вимоги до апаратної частини співпадають з вимогами з технічного завдання.

3.5.2 Програмна частина

Для виконання тестування апаратна платформа повинна мати операційну систему на базі macOS або іншу unix-сумісну, зі встановленою системою контейнеризації Docker.

3.5.3 Інструменти

Для виконання тестування використовувати наступні програмні інструменти:

– go test (Входить до набору розробки мови C#);

– GitHub;

– Vegeta– для тестування швидкодії.

ВИСНОВКИ ПО РОЗДІЛУ

У цьому розділі ми протестували систему за допомогою трьох видів тестування(компонентне, системне, продуктивності). Розроблений веб додаток успішно пройшов всі види тестування. Тестування застосунку у процесі розробки надає значні переваги. До них можна віднести наступне: швидке виявлення помилок і несправностей, рефакторинг коду, стабільна швидкість розробки, що веде до зменшення витрат. Тільки послідовно виконуючи всі тести, можна підтвердити правильність, функціональну поведінку та зручність використання застосунку, перш ніж публікувати його.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до складу і параметрів технічних засобів

4.1.1 Мінімальна конфігурація технічних засобів:

– Процесор з тактовою частотою не нижче 1,8 ГГц. Рекомендується використовувати як мінімум двоядерний процесор;

– 2 ГБ оперативної пам'яті; рекомендується 8 ГБ оперативної пам'яті (мінімум 2,5 ГБ при виконанні на віртуальній машині);

– Місце на жорсткому диску: до 210 ГБ (мінімум 800 МБ) вільного місця в залежності від встановлених компонентів; зазвичай для установки потрібно від 20 до 50 ГБ вільного місця;

– Швидкість жорсткого диска: для підвищення продуктивності встановіть операційну систему і Visual Studio на твердотільний накопичувач (SSD);

– Відеоадаптер з мінімальним дозволом 720p (1280 на 720 пікселів); для оптимальної роботи Visual Studio рекомендується дозвіл WXGA (1366 на 768 пікселів) або вище[12].

4.2 Розгортання програмного забезпечення

Для розгортання даного проекту потрібні наступні етапи:

– встановити SQL Server;

– встановити будь-який браузер;

– створити аккаунт для користувача SQL Server;

– встановити додаток VS для запуску програмного забезпечення;

– встановити в VS – entity framework;

– зробити build проекту.

4.3 Інструкція користувача замовника

4.3.1 Головна сторінка

На головній сторінці веб додатку користувач має можливість обрати, що саме йому потрібно – створити замовлення або виконати замовлення. Також користувач може одразу натистнути на посилання «Вход» - вхід в особистий кабінет або «Регистрация» та пройти реєстрацію(Рисунок 4.1).

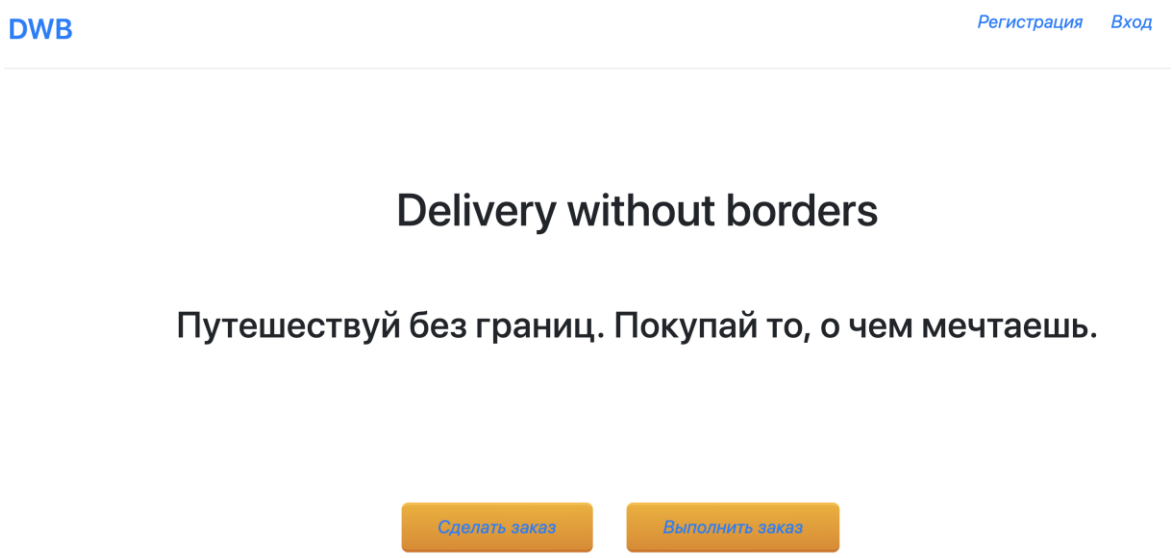

Рисунок 4.1 – Сторінка авторізації

4.3.2 Реєстрація

Для реєстрації, замовнику потрібно вказати наступну інформацію, а саме:

– електронну адресу;
– ім’я;
– прізвище;
– майбутній пароль від сторінки.

Після цього, вам потрібно обрати вашу роль реєстрації, вибравши один з трьох варіантів(в випадку замовника – потрібно вибрати «Клієнт») та натиснути кнопку «Регистрация». Після виконання всіх дій система поверне вас на головну сторінку на якій буде обмежено доступ до посилань, які не

зв’язані з замовником. Також с цієї сторінки є можливість повернутися на головну сторінку без реєстрації або перейти на посилання «Вход» (Рисунок 4.2).

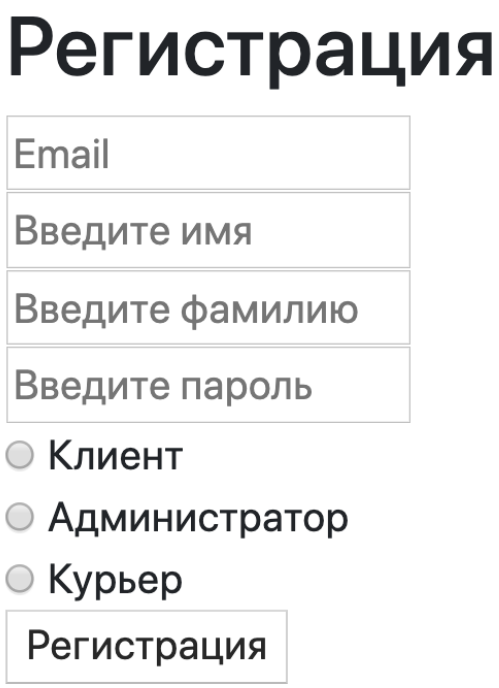

Рисунок 4.2 – Сторінка реєстрації для замовника

4.3.3 Вхід

Після успішної реєстрації замовник буде одразу авторизований, але якщо він реєструвався раніше, йому потрібно буде ввести електрону адресу та пароль, потім натиснути кнопку «Войти», після цього замовник перейде по посиланню на головну сторінку(Рисунок 4.3).

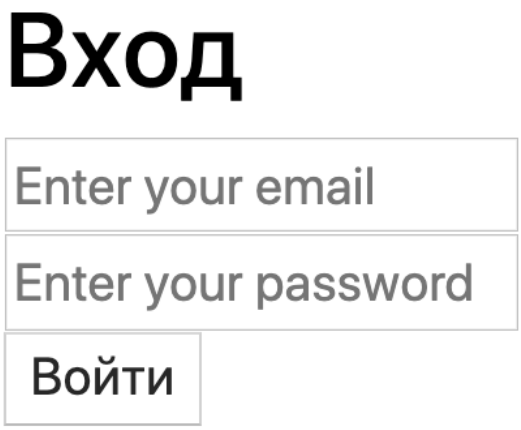

Рисунок 4.3 – Сторінка авторизації

4.3.4 Головна сторінка для замовника

Після авторизації, на головій сторінці, замовник має доступ до кнопки «Сделать заказ» та до посилань «Личный кабинет» і «Выход».Інші посилання для нього не доступні(Рисунок 4.4).

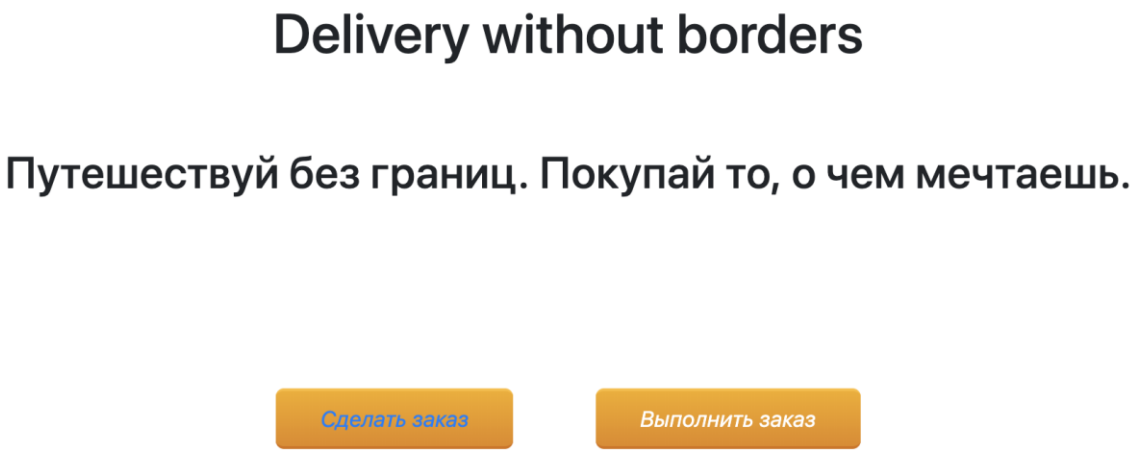

Рисунок 4.4 – Головна сторінка для замовника

4.3.5 Особистий кабінет для замовника

В особистому кабінеті замовник може переглядати інформацію про себе, керувати своїми замовленями та читати та відправляти повідомлення виконавцю. Також в особистому кабінеті можна переглянути баланс, зробити через посилання нове замовлення або вийти з авторизованої сторінки. В особистому кабінеті є посилання на головну сторінку. (Рисунок 4.5)

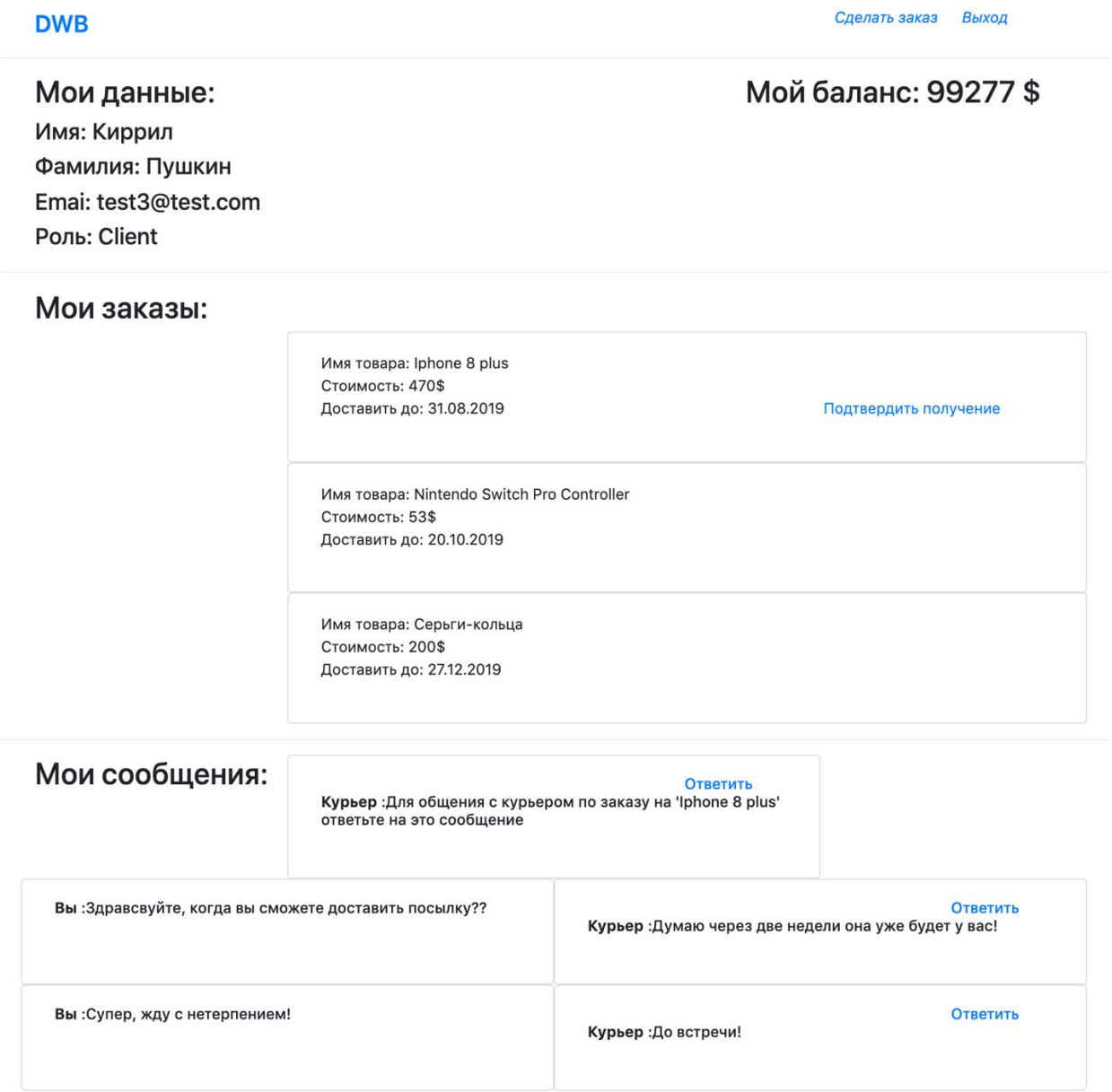

Рисунок 4.5 – Особиста сторінка замовника

4.3.6 Сторінка «Сделать заказ»

На цій сторінці, замовнику потрібно вставити посилання на товар, який він хоче замовити та натиснути кнопку «Отправить», після чого, замовник перейде на продовження створення заказу за посиланням. Також у нього є посилання на рекомендовані інтернет-магазини та на головну сторінку (Рисунок 4.6).

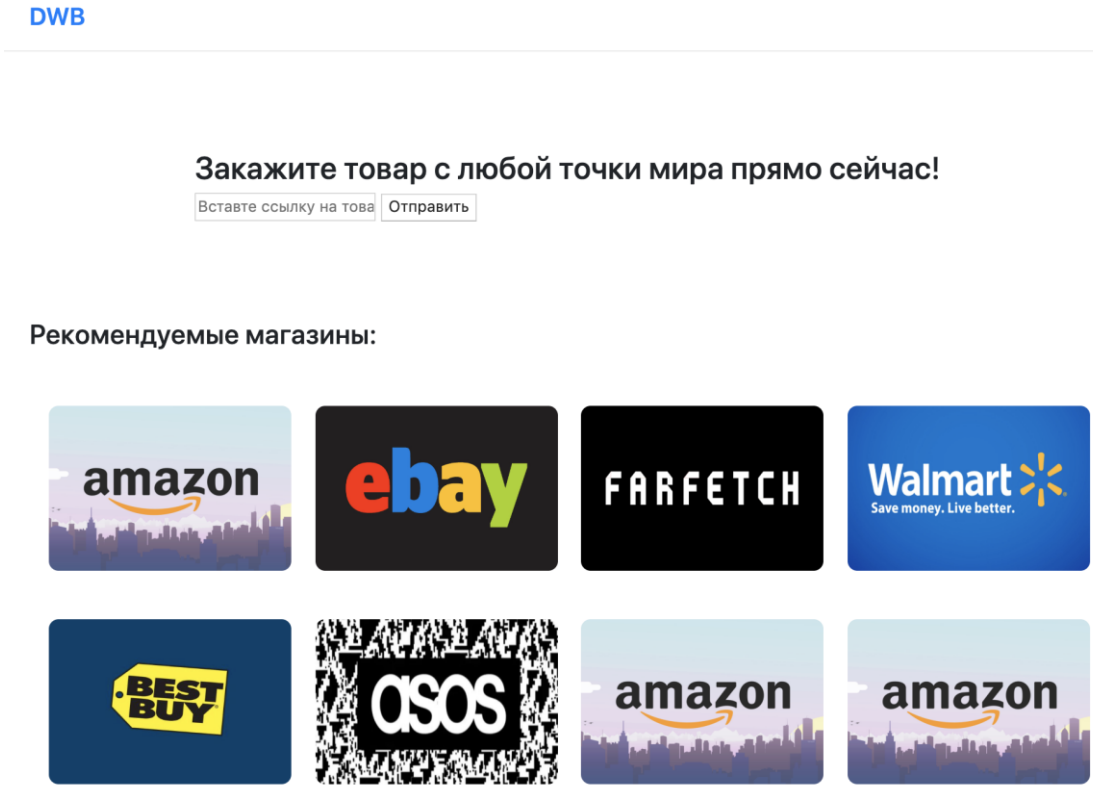

Рисунок 4.6 – Особиста сторінка замовника

4.3.7 Сторінка «Закончить заказ»

На цій сторінці замовник вказує інформацію про замовлення, після чого потрібно натиснути кнопку «Отправить». Після цього у замовника с баласну спишуться кошти, на ту суму, яку він вказав в заявці(Рисунок 4.7).

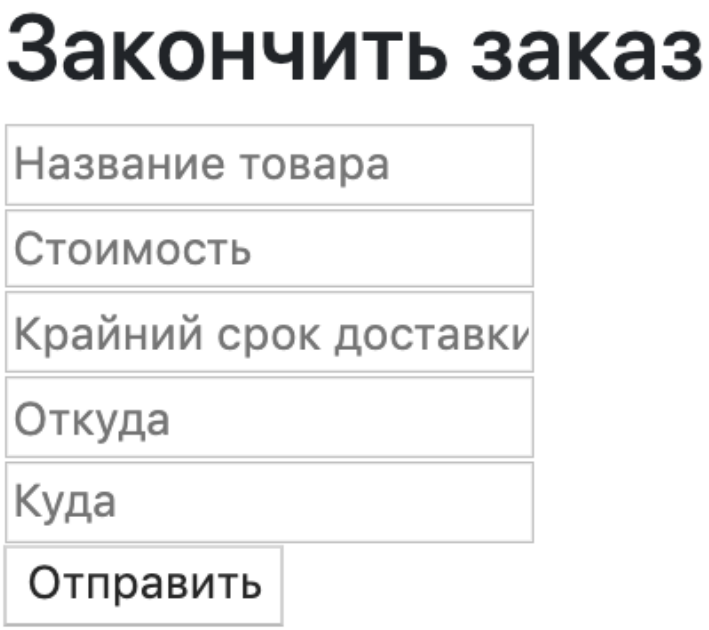

Рисунок 4.7 – Особиста сторінка замовника

4.4 Інструкція користувача виконавця

4.4.1 Реєстрація

Для реєстрації, виконавцю потрібно вказати наступну інфомацію, а саме:

– електронну адресу;
– ім’я;
– прізвище;
– майбутній пароль від сторінки.

Після цього, вам потрібно обрати вашу роль реєстрації, вибравши один з трьох варіантів(в випадку замовника – потрібно вибрати «Курьер») та натиснути кнопку «Регистрация». Після виконання всіх дій система поверне вас на головну сторінку на якій буде обмежено доступ до посилань, які не

зв’язані з кур’єром. Також с цієї сторінки є можливість повернутися на головну сторінку без реєстрації або перейти на посилання «Вход» (Рисунок 4.8).

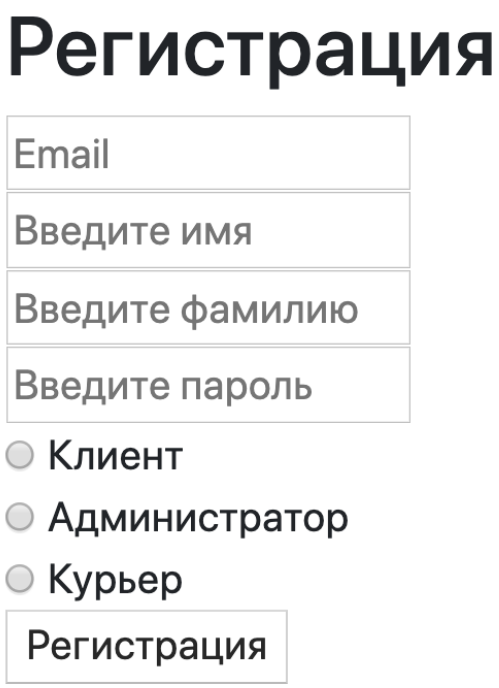

Рисунок 4.8 – Сторінка реєстрації для виконавця

4.4.2 Вхід

Після успішної реєстрації виконавець буде одразу авторизований, але якщо він реєструвався раніше, йому потрібно буде ввести електрону адресу та пароль, потім натиснути кнопку «Войти», після цього виконавець перейде по посиланню на головну сторінку(Рисунок 4.9).

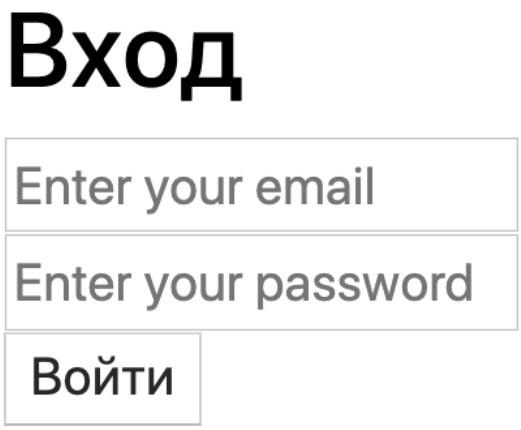

Рисунок 4.9 – Сторінка авторизації

4.4.3 Головна сторінка для виконавця

Після авторизації, на головій сторінці, виконавець має доступ до кнопки «Выполнить заказ» та до посилань «Личный кабинет» і «Выход». Інші посилання для нього не доступні(Рисунок 4.10).

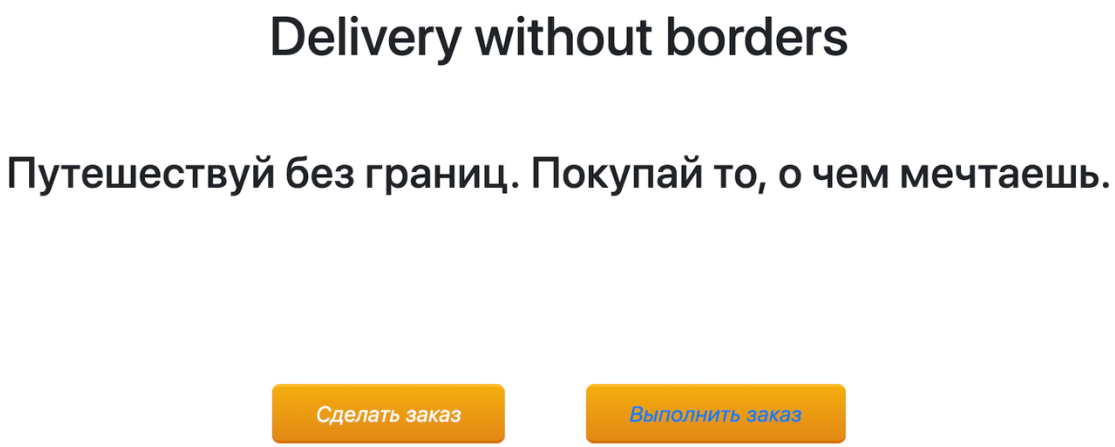

Рисунок 4.10 – Головна сторінка для виконавця

4.4.4 Особистий кабінет для виконавця

В особистому кабінеті виконавесь може переглядати інформацію про себе, керувати своїми замовленями та читати та відправляти повідомлення замовнику. Також в особистому кабінеті можна переглянути баланс, зробити через посилання нове замовлення або вийти з авторизованої сторінки. В особистому кабінеті є посилання на головну сторінку(Рисунок 4.11).

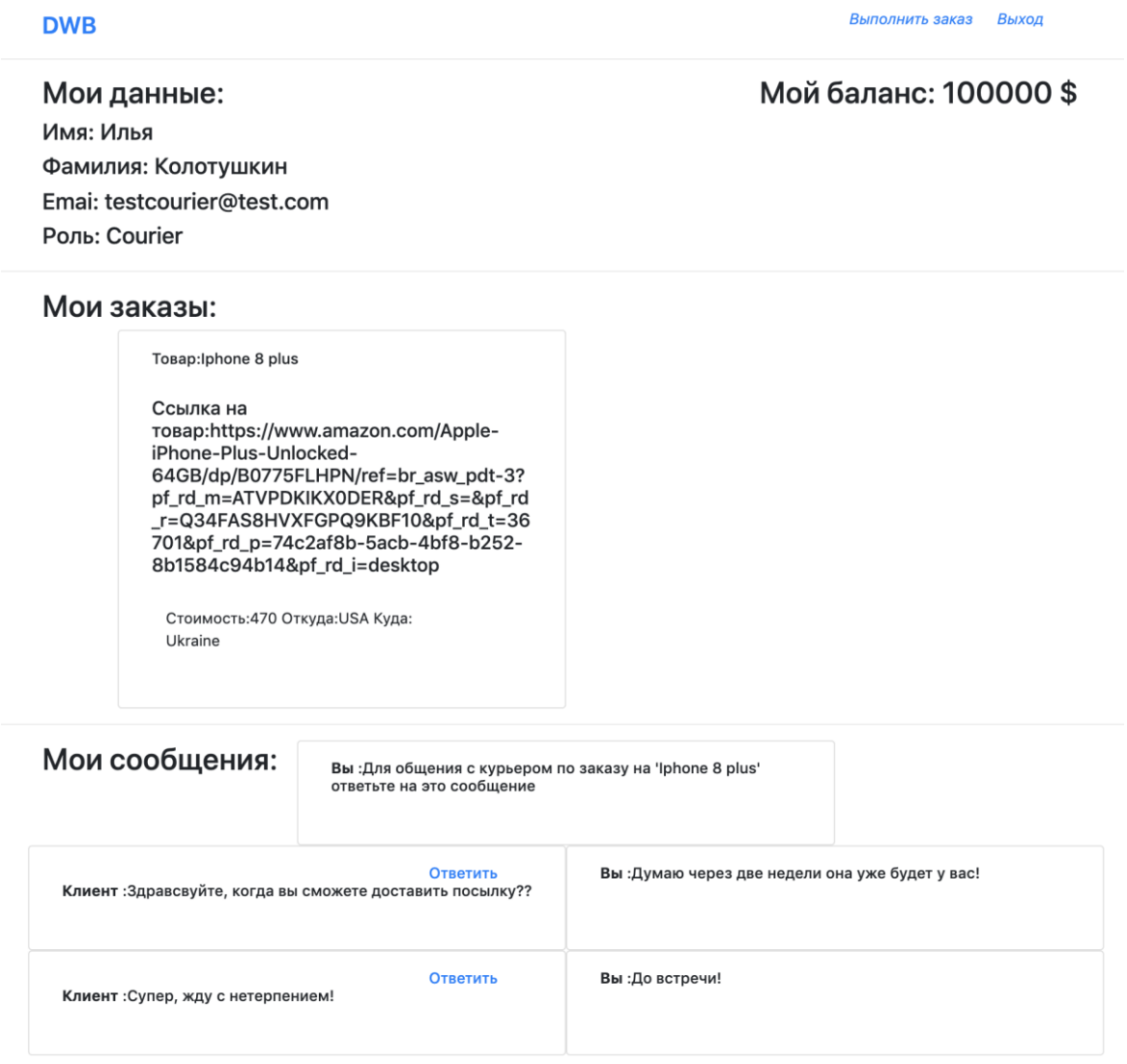

Рисунок 4.11 – Особиста сторінка виконавця

4.4.5 Сторінка «Выполнить заказ»

На цій сторінці, виконавець може вибрати товар, який він хоче доставити для замовника, йому доступна інформація про замовлення та посилання на замовлення. Щоб обрати замовлення, виконавцю потрібно натиснути кнопку «Выполнить», після цього замовлення буде додано до особистого кабінету виконавця(Рисунок 4.12).

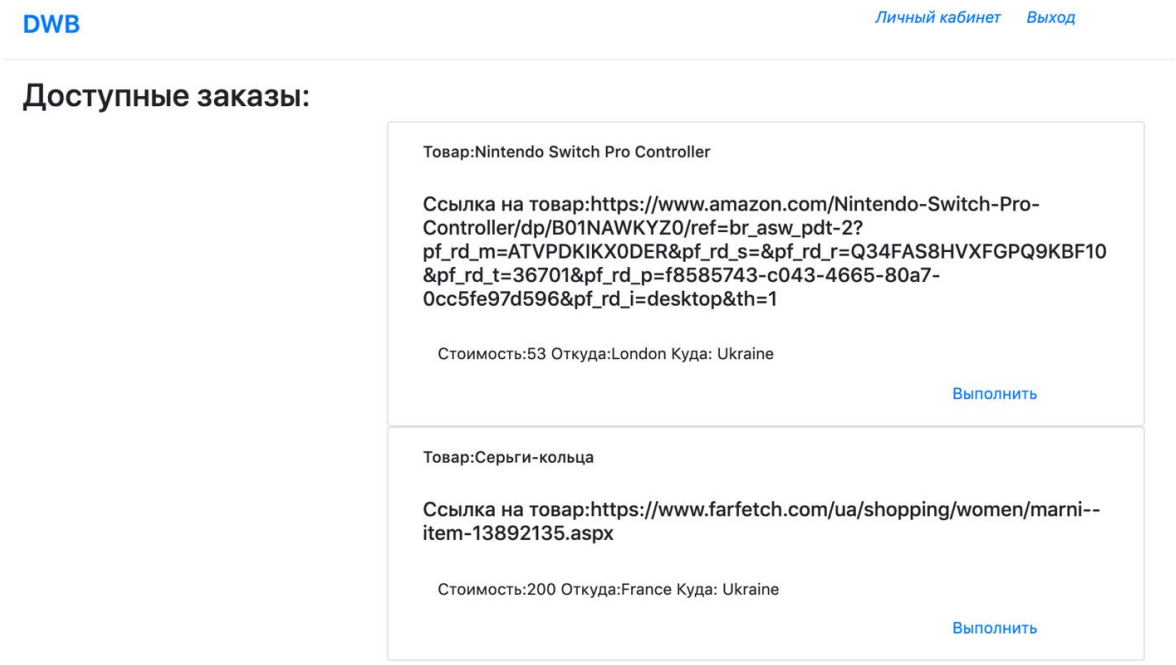

Рисунок 4.12 – Сторінка«Выполнить заказ»

4.5 Інструкція користувача адміністратора

4.5.1 Реєстрація

Для реєстрації, адміністратору потрібно вказати наступну інфомацію, а саме:

– електронну адресу;
– ім’я;
– прізвище;
– майбутній пароль від сторінки.

Після цього, вам потрібно обрати вашу роль реєстрації, вибравши один з трьох варіантів(в випадку замовника – потрібно вибрати «Администратор») та натиснути кнопку «Регистрация». Після виконання всіх дій система поверне вас на головну сторінку на якій буде обмежено доступ до посилань, які не зв’язані з адміністратором. Також с цієї сторінки є можливість повернутися на головну сторінку без реєстрації або перейти на посилання «Вход» (Рисунок 4.13).

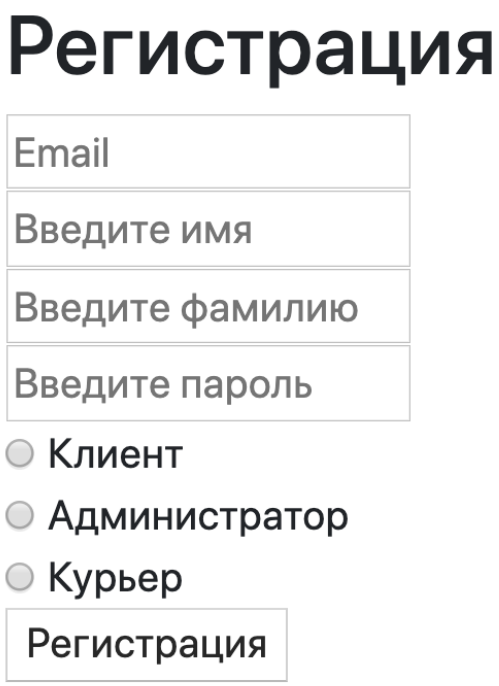

Рисунок 4.13 – Сторінка реєстрації для адміністратора

4.5.2 Вхід

Після успішної реєстрації адміністратор буде одразу авторизований, але якщо він реєструвався раніше, йому потрібно буде ввести електрону адресу та пароль, потім натиснути кнопку «Войти», після цього виконавець перейде по посиланню на головну сторінку(Рисунок 4.14).

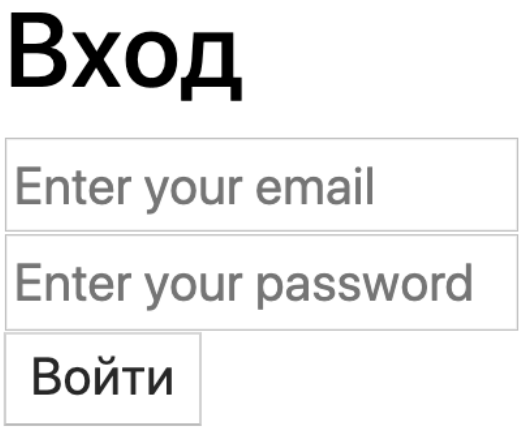

Рисунок 4.14 – Сторінка авторизації

4.5.3 Головна сторінка для адміністратора

Після авторизації, на головій сторінці, виконавець має доступ до посилань «Личный кабинет» і «Выход». Інші посилання для нього не доступні(Рисунок 4.15).

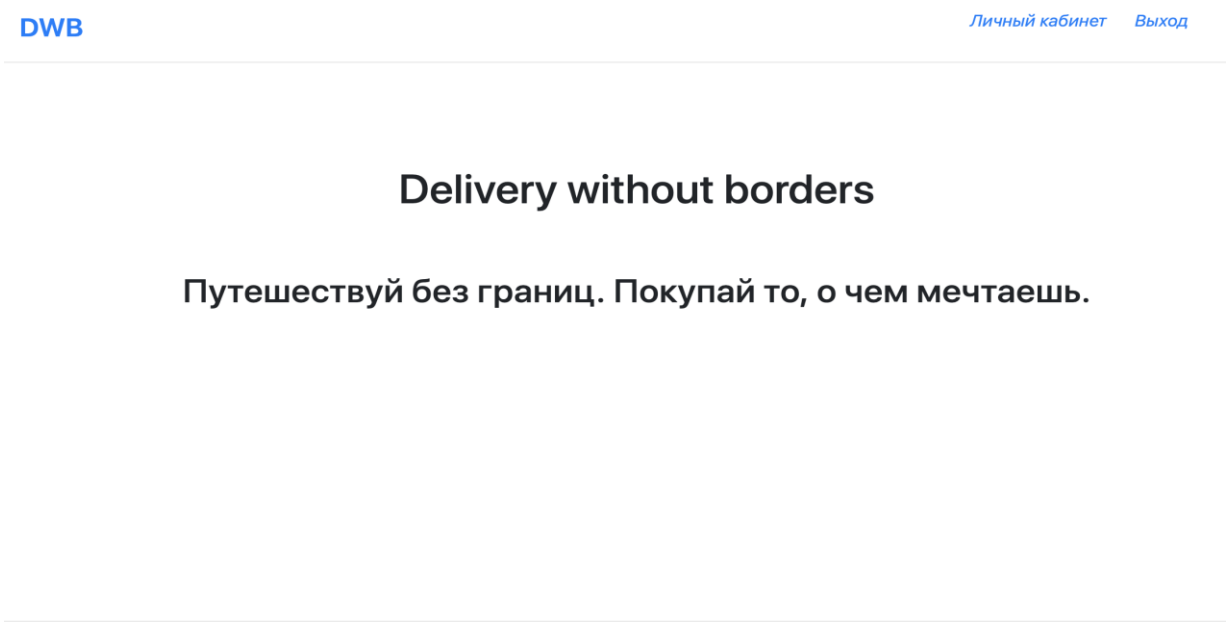

Рисунок 4.15 – Головна сторінка для адміністратора

4.5.4 Особистий кабінет для адміністратора

В особистому кабінеті адміністратор може переглядати інформацію про себе, керувати замовленнями та читати та відправляти повідомлення клієнтам. Також в особистому кабінеті можна переглянути баланс, зробити через посилання нове замовлення або вийти з авторизованої сторінки. В особистому кабінеті є посилання на головну сторінку(Рисунок 4.16).

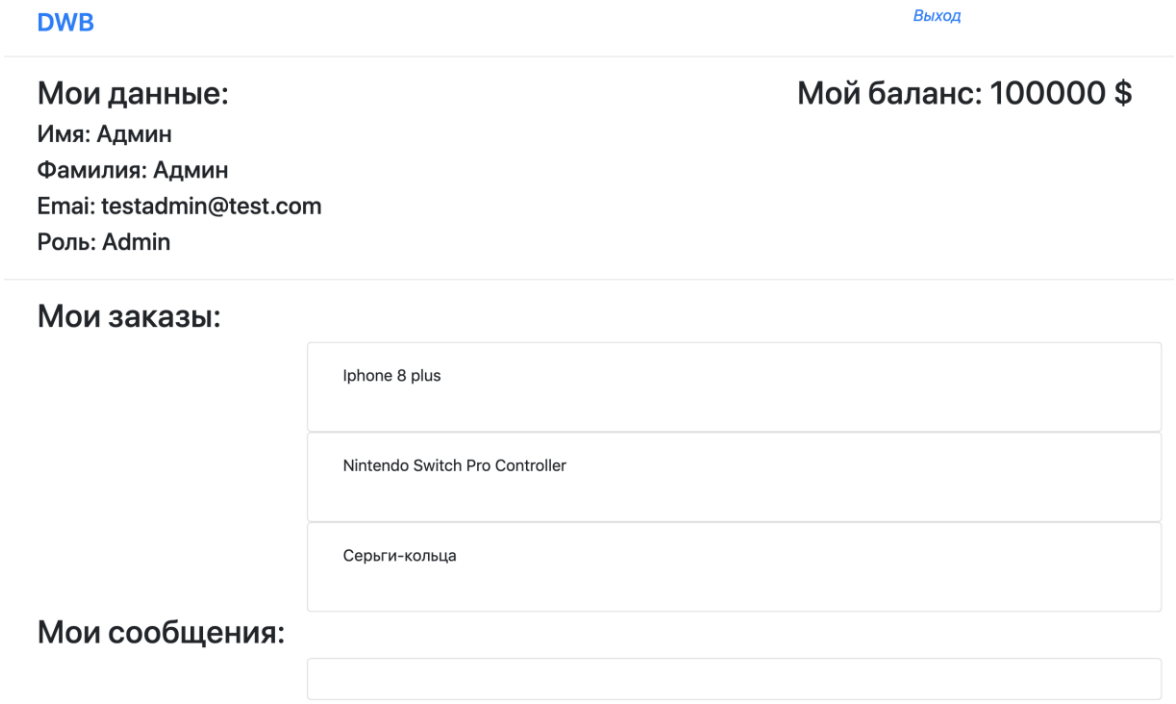

Рисунок 4.16 – Особиста сторінка адміністратора

ВИСНОВОК ПО РОЗДІЛУ

У цьому розділі були визначені мінімальні характеристика технічних засобів, також було описано, що потрібно для розгортання програмного забезпечення. Розроблено детельну інструкцію користувача та адміністратора. Через те, що ця система розрахована на два види користувачів та адміністратора, у розділі подано три інструції користувача.

ВИСНОВКИ

У ході розробки дипломного проекту було проведено аналіз вимог до програмного забезпечення. Були проаналізовані реалізовані існуючі технічні рішення, які працюють с доставкою товарів за допомогою мандрівників. Було знайдено недоліки у діючих технічних рішеннях. Також були сформовані вимоги до веб застосунку.

Розроблений веб застосунок має ряд переваг на існуючими аналогами:

– користувачу(виконавцю) не потрібно чекати жодних підтверджень, він може одразу братись за виконання завдання;

– користувачі можуть тримати зв’язок за допомоги листування у веб застосунку;

– у веб застосунку, усі замовлення реальні та виконувані, тому що перед публікацією вони потрапляють на перевірку до адміністратора.

Після того, як було завершено етап розробки, було проведено тестування та виправлено декілька недоліків.

Також було розроблено проектну документацію, описано усі класи, побудовано схему класів програмного забезпечення, схему бізнес-процесів веб застосунку, тестування та керівництво користувача та адміністратора.

ПЕРЕЛІК ПОСИЛАНЬ

1. Bootstrap [Електронний ресурс] – Режим доступу до ресурсу: https://getbootstrap.ru/.
2. What is jQuery? [Електронний ресурс] – Режим доступу до ресурсу: https://jquery.com/
3. MVC — модель-представление-контроллер [Електронний ресурс] – Режим доступу до ресурсу: https://web-creator.ru/articles/mvc/.
4. Введение в Entity Framework [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/sharp/entityframework/1.1.php/.
5. Технологии баз данных: SQL, T-SQL, PL/SQL, реляционные БД [Електронний ресурс] – Режим доступу до ресурсу: http://datasql.ru/baseworkbd/8.html/.
6. Понимая Docker [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/253877/.
7. Рівні тестування [Електронний ресурс] – Режим доступу до ресурсу: http://qalearning.com.ua/theory/lectures/material/testing-levels/.
8. Качество программного обеспечения[Електронний ресурс] – Режим доступу до ресурсу: https://studme.org/128171/informatika/kachestvo_programmnogo_obespecheniya.
9. Что такое TDD и модульное тестирование [Електронний ресурс] – Режим доступу до ресурсу: https://javarush.ru/groups/posts/6-chto-takoe-tdd-i-moduljhnoe-testirovanie-.
10. ЛР №8 Системное тестирование [Електронний ресурс] – Режим доступу до ресурсу: http://www.4stud.info/software-construction-and-testing/work8.html.
11. Нагрузочное тестирование vs Тестирование производительности [Електронний ресурс] – Режим доступу до ресурсу: https://www.performance-lab.ru/blog/load-testing/testirovanie-proizvoditelnosti/.

12. Visual Studio 2019 Product Family System Requirements [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/visualstudio/releases/2019/system-requirements/.

13. SQL за 20 минут [Електронний ресурс] – Режим доступу до ресурсу:https://proglib.io/p/sql-for-20-minutes/.

14. Visual Studio Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/visualstudio/?view=vs-2019.

15. Примеры и рекомендации удобных инструкций [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/153973/.

ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ

Все эти листы не печатаются! Они нужны только для того, чтоб сделать правильное содержание

ДОДАТОК Б ОПИС ПРОГРАМИ

ДОДАТОК В ПРОГРАМА ТА МЕТОДИКА ТЕСТУВАННЯ

Все эти листы не печатаются! Они нужны только для того, чтоб сделать правильное содержание

ДОДАТОК Г КЕРІВНИЦТВО СИСТЕМНОГО ПРОГРАМІСТА

ДОДАТОК Д КЕРІВНИЦТВО ПРОГРАМІСТА

ДОДАТОК Є КЕРІВНИЦТВО КОРИСТУВАЧА

ДОДАТОК Е ГРАФІЧНИЙ МАТЕРІАЛ

ЛИСТ 1. СХЕМА СТРУКТУРНА ВАРІАНТІВ ВИКОРИСТАНЬ

ЛИСТ 2. СХЕМА СТРУКТУРНА СТАНІВ СИСТЕМИ

ЛИСТ 3. СХЕМА БАЗИ ДАНИХ

ЛИСТ 4. СХЕМА СТРУКТУРНА КЛАСІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ЛИСТ 5. КРЕСЛЕННЯ ВИГЛЯДУ ЕКРАННИХ ФОРМ